SC22

Dallas, TX | hpc accelerates.

Accelerating Datalog Applications with cuDF

Authors: Ahmedur Rahman Shovon, Landon Richard Dyken, Oded Green, Thomas Gilray, Sidharth Kumar

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

NVIDIA

Outline

- Introduction
- Contributions
- Experimental Setup & Dataset
- Results
- Limitations
- Future work

Datalog: a bottom-up logic programming language

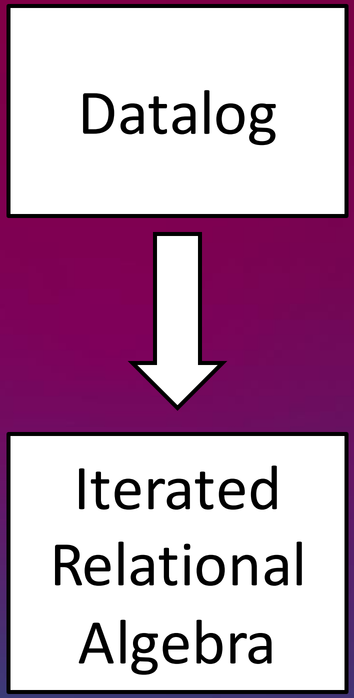
A lightweight logic-programming language for deductive-database systems

Input Database → **Datalog (set of rules)** → **Output Database**

Running the Datalog program extends data from input database creating the output database with all data transitively derivable via the program rules

- Ceri, S., Gottlob, G., & Tanca, L. (1989). What you always wanted to know about Datalog (and never dared to ask). IEEE transactions on knowledge and data engineering, 1(1), 146-166.
- Gilray, T., Kumar, S., & Micinski, K. (2021, March). Compiling data-parallel datalog. In Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction (pp. 23-35).

Bottom-up logic programming with Datalog



Datalog rule for computing transitive closure

```
T(x,y) <- G(x,y).
T(x,z) <- T(x,y), G(y,z).
```

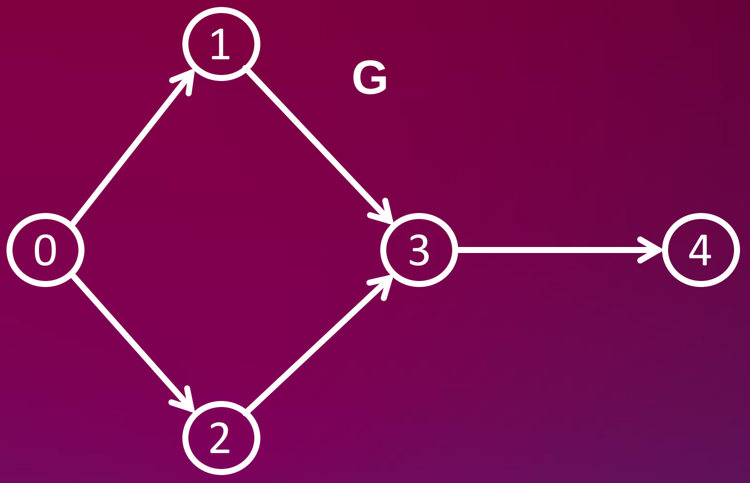
Operationalized as a fixed-point iteration using F_G

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

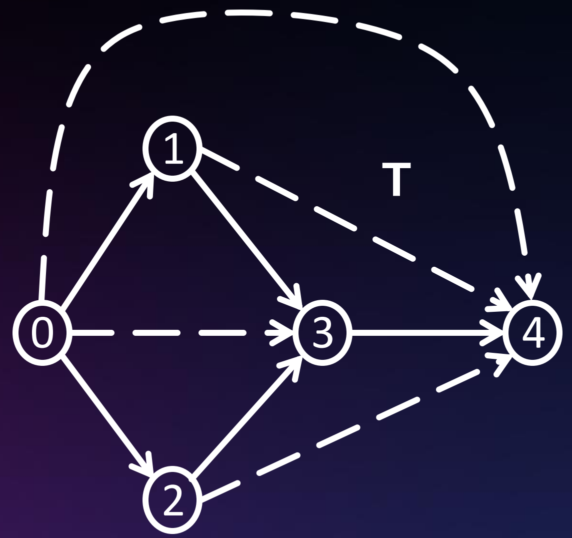
Relational algebra: **Union** **Projection** **Join**

- Gilray, T., & Kumar, S. (2019, December). Distributed relational algebra at scale. In 2019 IEEE 26th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 12-22). IEEE.
- Kumar, S., & Gilray, T. (2020, June). Load-balancing parallel relational algebra. In International Conference on High Performance Computing (pp. 288-308). Springer, Cham.

Transitive Closure: Logical Inference for Graphs



T (x, y) <- G (x, y)
T (x, z) <- T (x, y), G (y, z)

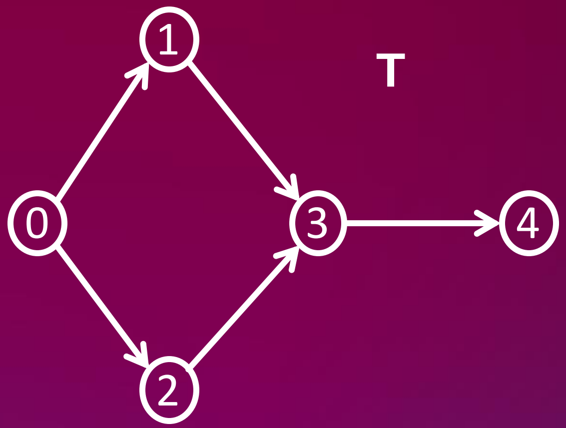


0	1
1	3
3	4
0	2
2	3

0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

$$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

Transitive Closure: Iterations

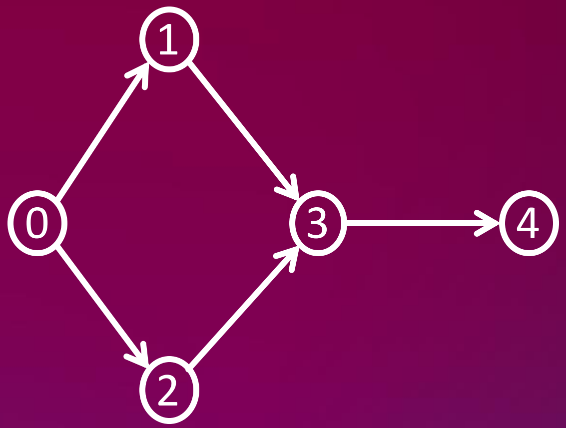


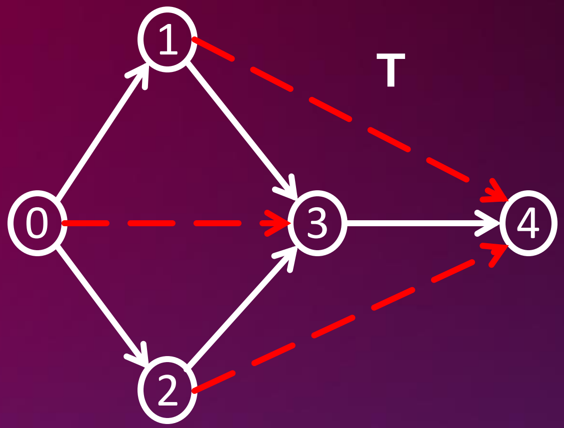
T

0	**1**
0	**2**
1	**3**
2	**3**
3	**4**

$$1 \quad F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

Transitive Closure: Iterations



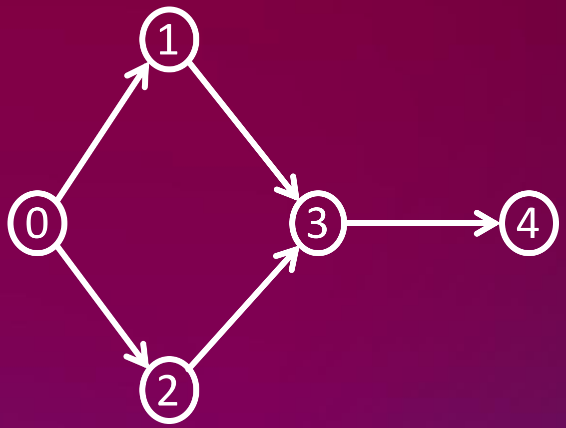


0	1
0	2
1	3
2	3
3	4

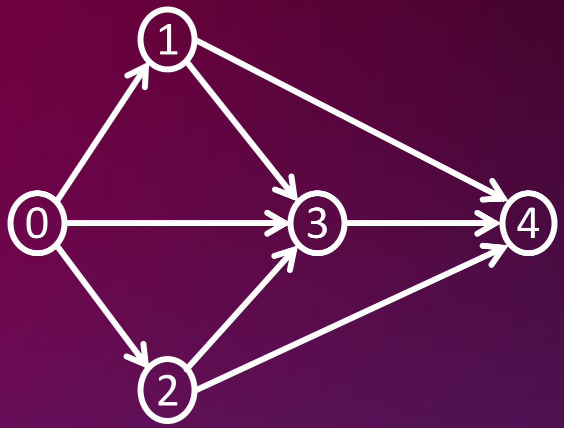
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4

2 $F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$

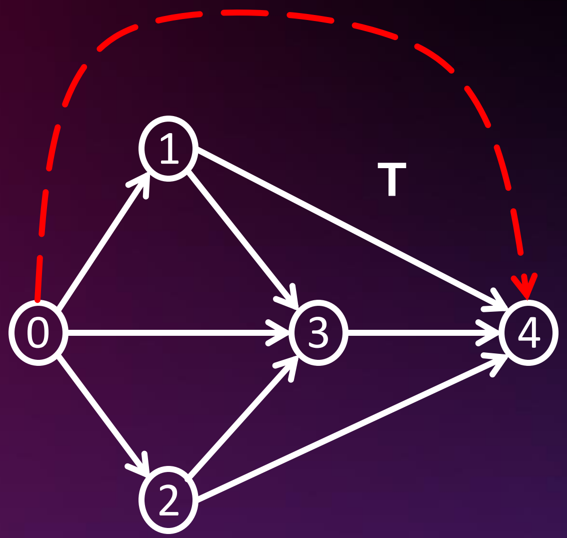
Transitive Closure: Iterations



0	1
0	2
1	3
2	3
3	4



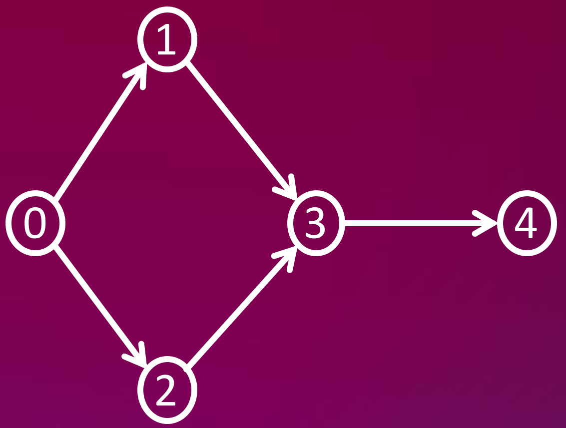
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4



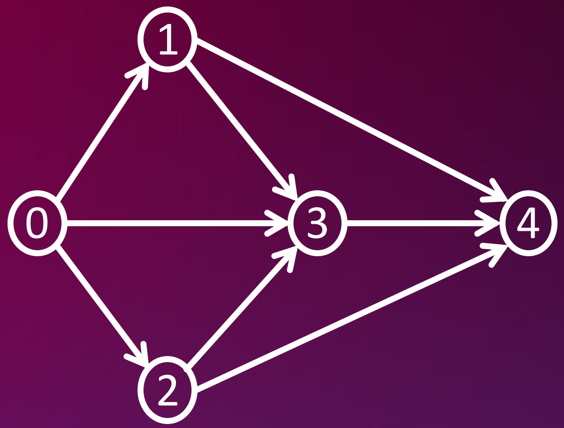
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

3 $$F_G(T) \triangleq G \cup \Pi_{1,2}(\rho_{0/1}(T) \bowtie_1 G)$$

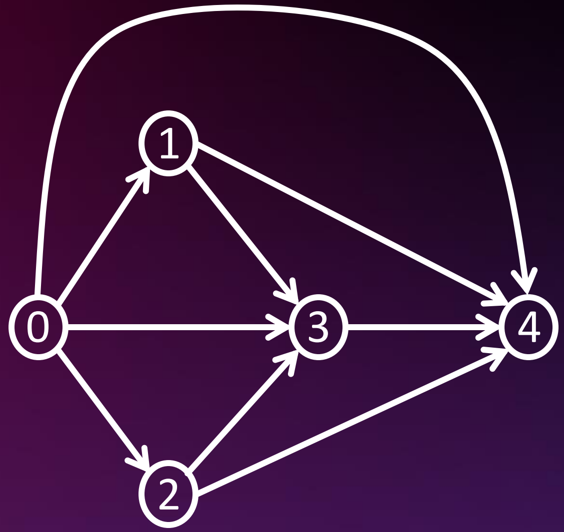
Transitive Closure: Iterations



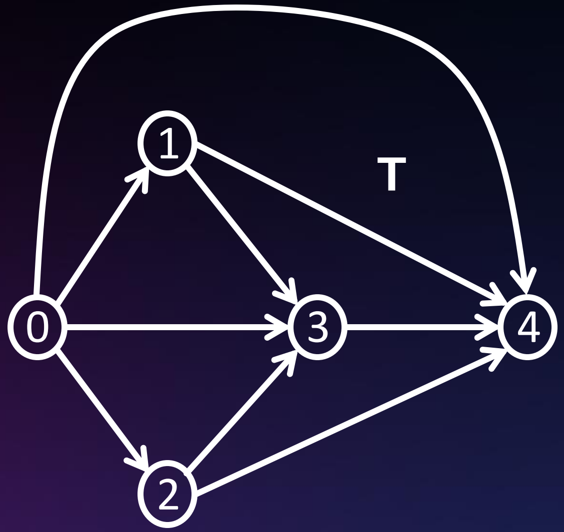
0	1
0	2
1	3
2	3
3	4



0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4



0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4



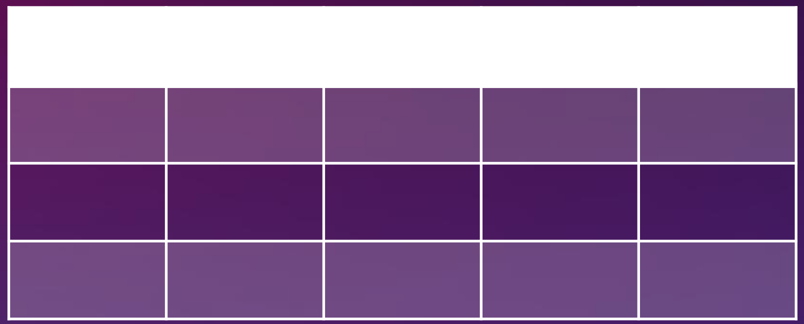
0	1
0	2
1	3
2	3
3	4
0	3
1	4
2	4
0	4

Algorithm for Transitive Closure computation using RA

```
procedure TRANSITIVECLOSURE(Graph G)
    result ← G
    R ← Rename(G)
    do
        newTriplets ← Join(R, G)
        inferredPaths ← Deduplication(Projection(newTriplets))
        oldLength ← Length(result)
        result ← Deduplication(Union(result, inferredPaths))
        currentLength ← Length(result)
        R ← Rename(inferredPaths)
    while oldLength ≠ currentLength
    return result
end procedure
```

DataFrame, Pandas, and cuDF

DataFrame: 2D labeled tabular data structure



DataFrame has RA primitives (e.g. join, aggregation, rename, deduplication, and projection)

- Reback, J., McKinney, W., Van Den Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 5. Zenodo.
- Chen, D. Y. (2017). Pandas for everyone: Python data analysis. Addison-Wesley Professional.
- Green, O., Du, Z., Patel, S., Xie, Z., Liu, H., & Bader, D. A. (2021, December). Anti-Section Transitive Closure. In 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 192-201). IEEE.
- Fender, A., Rees, B., & Eaton, J. RAPIDS cuGraph. In Massive Graph Analytics (pp. 483-493). Chapman and Hall/CRC.

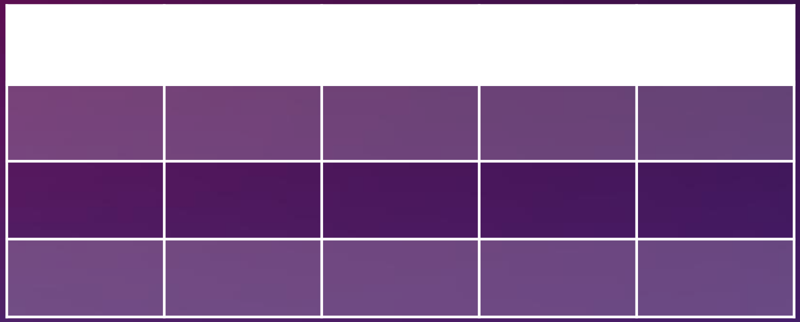
DataFrame, Pandas, and cuDF

pandas

RAPIDS

DataFrame: 2D labeled tabular data structure

CPU



GPU (NVIDIA CUDA)

DataFrame has RA primitives (e.g. join, aggregation, rename, deduplication, and projection)

- Reback, J., McKinney, W., Van Den Bossche, J., Augspurger, T., Cloud, P., Klein, A., ... & Seabold, S. (2020). pandas-dev/pandas: Pandas 1.0. 5. Zenodo.
- Chen, D. Y. (2017). Pandas for everyone: Python data analysis. Addison-Wesley Professional.
- Green, O., Du, Z., Patel, S., Xie, Z., Liu, H., & Bader, D. A. (2021, December). Anti-Section Transitive Closure. In 2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC) (pp. 192-201). IEEE.
- Fender, A., Rees, B., & Eaton, J. RAPIDS cuGraph. In Massive Graph Analytics (pp. 483-493). Chapman and Hall/CRC.

Outline

- Introduction
- Contributions
- Experimental Setup & Dataset
- Results
- Limitations
- Future work

Contributions

Demonstrated the feasibility of implementing Datalog applications using the RA primitives of Pandas and cuDF

Contributions

Demonstrated the feasibility of implementing Datalog applications using the RA primitives of Pandas and cuDF

Implemented triangle counting and transitive closure using RA primitives of Pandas and cuDF

Algorithm for Transitive Closure computation using RA

procedure TRANSITIVECLOSURE(Graph G)
  $result \leftarrow G$
  $R \leftarrow \text{Rename}(G)$
  do
    $newTriplets \leftarrow \text{Join}(R, G)$
    $inferredPaths \leftarrow \text{Deduplication}(\text{Projection}(newTriplets))$
    $oldLength \leftarrow \text{Length}(result)$
    $result \leftarrow \text{Deduplication}(\text{Union}(result, inferredPaths))$
    $currentLength \leftarrow \text{Length}(result)$
    $R \leftarrow \text{Rename}(inferredPaths)$
  while $oldLength \neq currentLength$
  return $result$
end procedure

Implemented using cuDF and Pandas

Contributions

Demonstrated the feasibility of implementing Datalog applications using the RA primitives of Pandas and cuDF

Implemented triangle counting and transitive closure using RA primitives of Pandas and cuDF

147x speedup for triangle counting and 71x speedup for transitive closure computation using cuDF over Pandas

Contributions

Demonstrated the feasibility of implementing Datalog applications using the RA primitives of Pandas and cuDF

Implemented triangle counting and transitive closure using RA primitives of Pandas and cuDF

147x speedup for triangle counting and 71x speedup for transitive closure computation using cuDF over Pandas

Identified the shortcomings of Pandas and cuDF based Datalog implementations

Outline

- Introduction
- Contributions
- Experimental Setup & Dataset
- Results
- Limitations
- Future work

Experimental Setup

ThetaGPU supercomputer from Argonne Leadership Computing Facility

GPU: NVIDIA A100, CUDA: 11.4

CPU: AMD EPYC 7742 64-Core Processor

Operating System: Ubuntu 20.04 LTS

Python package information:

- Python version: 3.9.13
- Conda version: conda 4.13.0
- cuda-python: 11.7.0
- cudatoolkit: 11.2.72
- cudf: 22.06.01
- pandas: 1.4.3

➢ Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Dataset: Stanford large network dataset collection

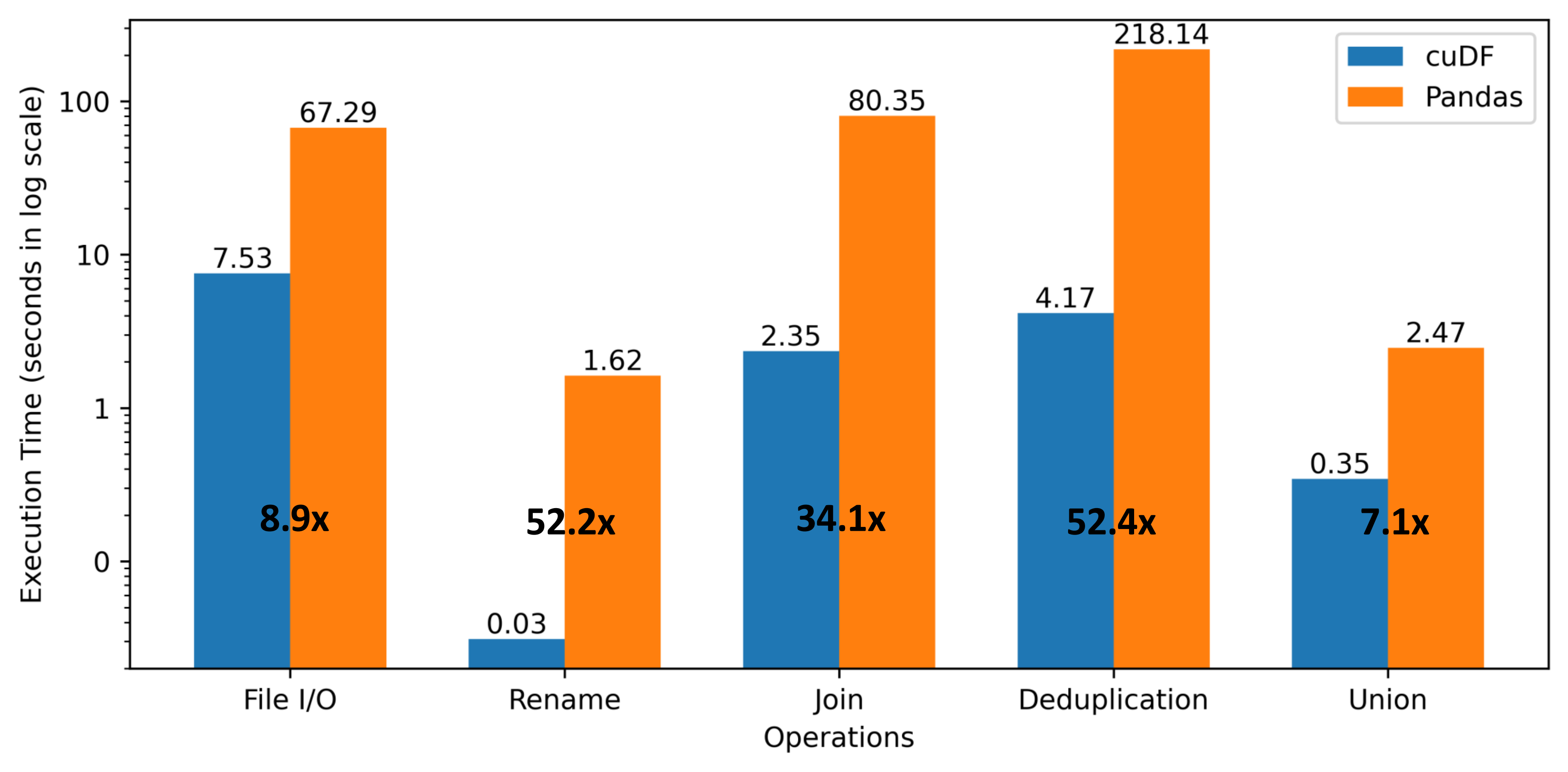
Graph	Type	Nodes	Edges
p2p-Gnutella09	Directed	8,114	26,013
p2p-Gnutella04	Directed	10,876	39,994
Skitter	Undirected	1,696,415	11,095,298
roadNet-CA	Undirected	1,965,206	2,766,607
roadNet-TX	Undirected	1,379,917	1,921,660
roadNet-PA	Undirected	1,088,092	1,541,898
SF.cedge	Undirected	1,74,955	2,23,001
cal.cedge	Undirected	21,048	21,693
TG.cedge	Undirected	18,263	23,874
OL.cedge	Undirected	6,105	7,035

➢ Leskovec, J., & Krevl, A. (2014). SNAP Datasets: Stanford large network dataset collection.

Outline

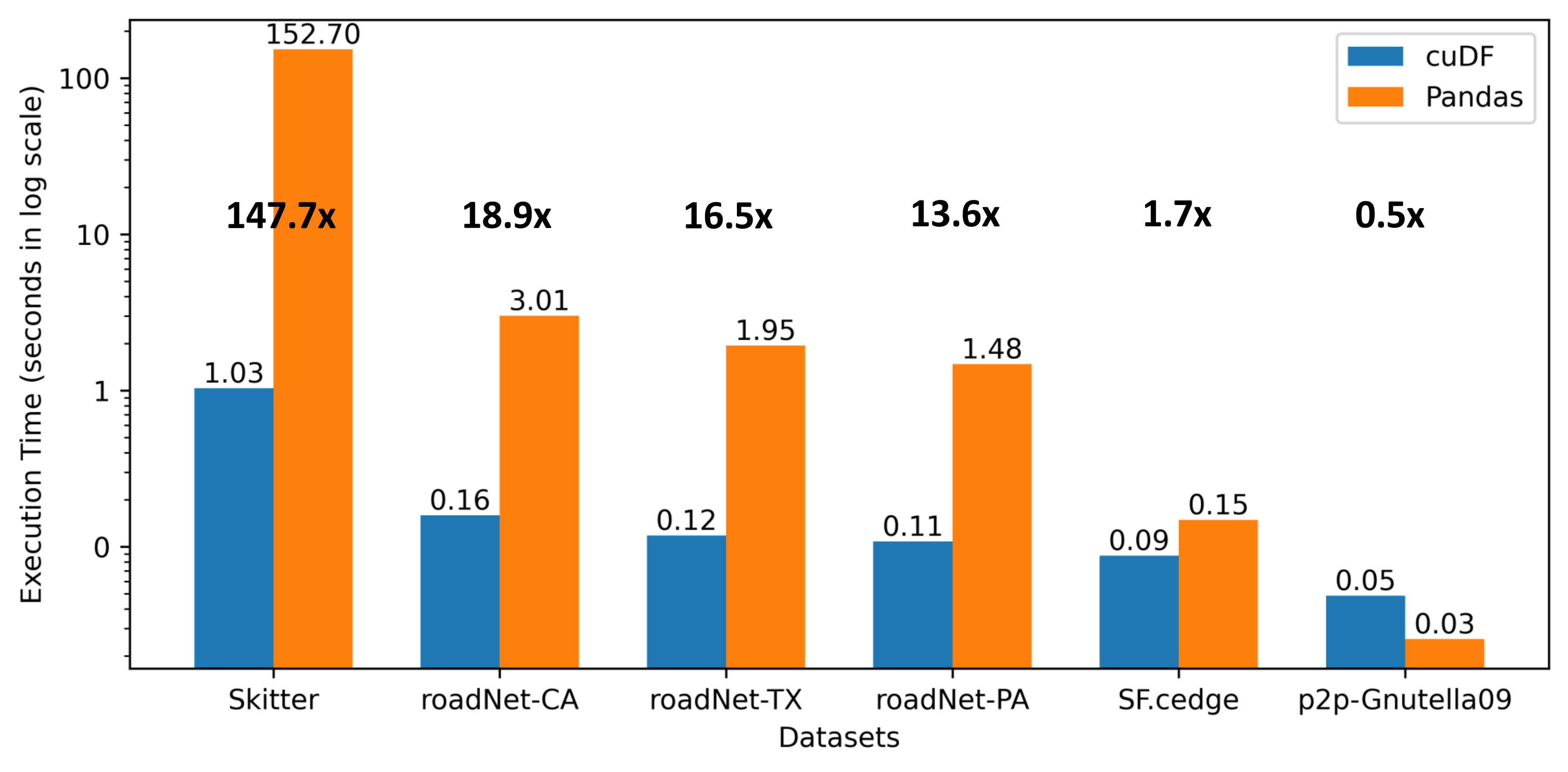
- Introduction
- Contributions
- Experimental Setup & Dataset
- Results
- Limitations
- Future work

Standalone relational operations



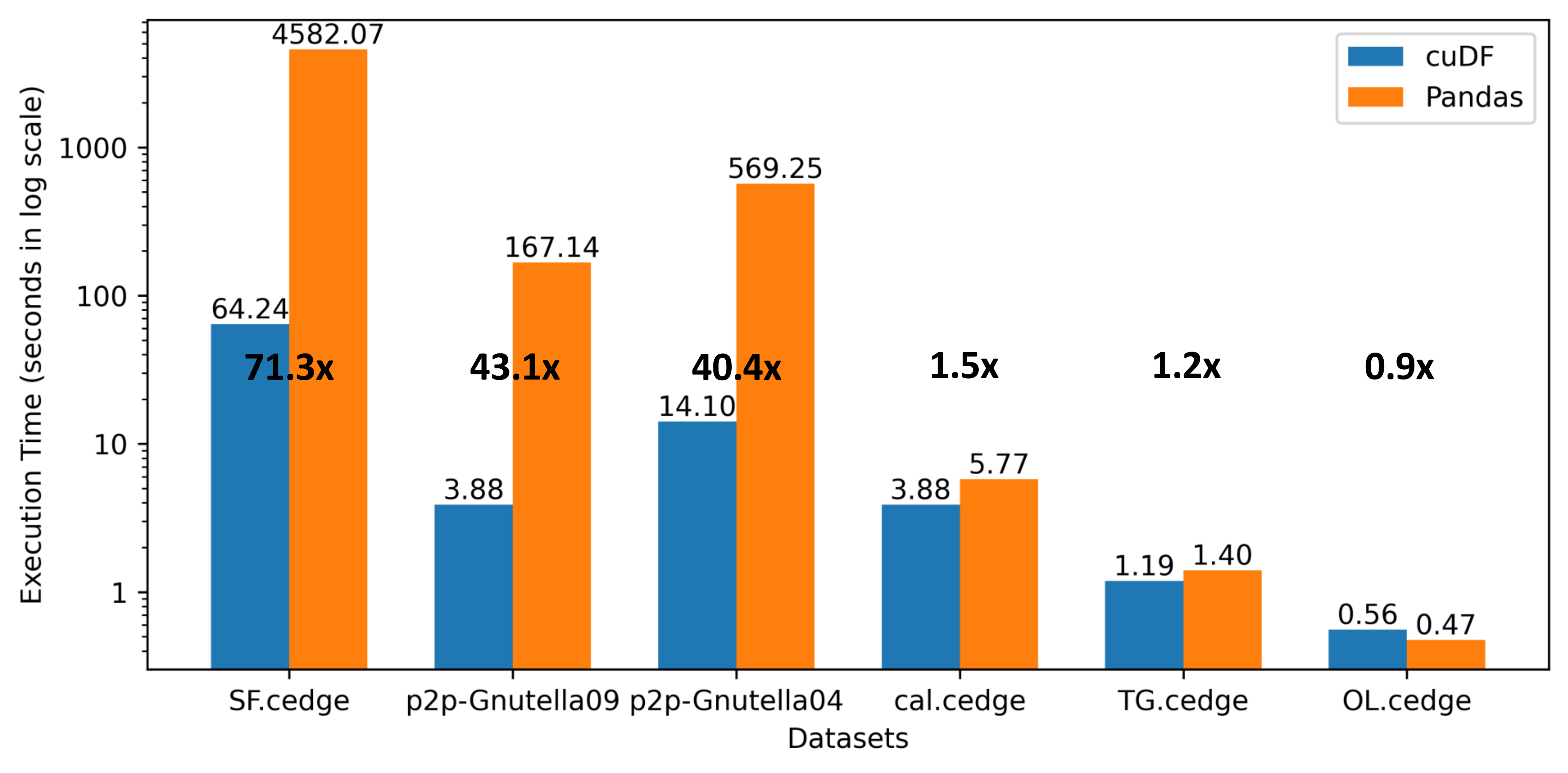
- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Triangle counting



- Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Transitive closure



➢ Leadership Computing Facility, A. (2022). Argonne Leadership Computing Facility. Theta GPU Nodes. URL: https://www.alcf.anl.gov/support-center/theta-gpu-nodes

Outline

- Introduction
- Contributions
- Experimental Setup & Dataset
- Results
- Limitations
- Future work

Limitations

- **No fusing**: Pandas and cuDF does not support fusing of RA operations
- **Memory and computation overhead**: Sequentially API calls require storing the intermediate values
- **Improvement idea**: Consecutive joins in triangle counting can be performed in one single operation
- **Memory overflow error**: Cannot perform TC computation for graphs with several million edges in cuDF

Outline

- Introduction
- Contributions
- Experimental Setup & Dataset
- Results
- Limitations
- Future work

Future work

Compare	Compare our Pandas solution with NetworkX Compare our cuDF solution with cuGraph and Hornet
Extend	Extend our solution to using Dask/cuDF for multi-GPU applications
Identify	Identify missing algorithmic components in Pandas and cuDF necessary for a broader set of applications for scalable Datalog backend

- Team, R. D. (2018). RAPIDS: Collection of libraries for end to end GPU data science. NVIDIA, Santa Clara, CA, USA. https://rapids.ai
- Hagberg, A., Swart, P., & S Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX. Los Alamos National Lab.(LANL), Los Alamos, NM (United States).
- Busato, F., Green, O., Bombieri, N., & Bader, D. A. (2018, September). Hornet: An efficient data structure for dynamic sparse graphs and matrices on gpus. In 2018 IEEE High Performance extreme Computing Conference (HPEC) (pp. 1-7). IEEE.

THANK YOU

https://github.com/harp-lab/gpujoin